

NETCONF Working Group K.  
Watsen  
Internet-Draft S.  
Hanna  
Intended status: Standards Track Juniper  
Networks  
Expires: January 02, 2015 J.  
Clarke  
Cisco  
Systems  
M.  
Abrahamsson  
T-  
Systems  
July 1,  
2014

**Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)  
draft-ietf-netconf-zerotouch-00**

Abstract

This draft presents a technique for establishing a secure NETCONF connection between a newly deployed IP-based device, configured with just its factory default settings, and the new owner's Network Management System (NMS).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 02, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with

respect  
to this document. Code Components extracted from this document must

Watsen, et al.  
1]

Expires January 02, 2015

[Page

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

<u>1.</u>	Introduction . . . . .	
<u>3</u>		
<u>1.1.</u>	Terminology . . . . .	
<u>3</u>		
<u>1.2.</u>	Objectives . . . . .	
<u>3</u>		
<u>1.3.</u>	Use Cases . . . . .	
<u>4</u>		
<u>1.4.</u>	Actors and Roles . . . . .	
<u>5</u>		
<u>2.</u>	Configuration Server . . . . .	
<u>7</u>		
<u>2.1.</u>	Service Interface . . . . .	
<u>7</u>		
<u>2.2.</u>	Interactive Interface . . . . .	
<u>7</u>		
<u>2.3.</u>	Transport Security . . . . .	
<u>8</u>		
<u>2.4.</u>	Expiration Policy . . . . .	
<u>8</u>		
<u>2.5.</u>	Troubleshooting and Auditing . . . . .	
<u>8</u>		
<u>3.</u>	Configuration Signer . . . . .	
<u>9</u>		
<u>3.1.</u>	Overview . . . . .	
<u>9</u>		
<u>3.2.</u>	Signing Configurations . . . . .	
<u>9</u>		
<u>3.3.</u>	Optional Encryption . . . . .	
<u>9</u>		
<u>3.4.</u>	Delegation of Trust . . . . .	
<u>9</u>		
<u>3.5.</u>	Delegation to a Specific Customer . . . . .	
<u>10</u>		
<u>4.</u>	Device . . . . .	
<u>10</u>		
<u>4.1.</u>	Overview . . . . .	
<u>10</u>		
<u>4.2.</u>	Factory Default State . . . . .	
<u>11</u>		
<u>4.3.</u>	Boot Sequence . . . . .	
<u>12</u>		
<u>5.</u>	Network Management System (NMS) . . . . .	
<u>15</u>		
<u>5.1.</u>	Overview . . . . .	
<u>15</u>		

<a href="#">16</a>	<a href="#">5.2.</a>	Precondition . . . . .
<a href="#">17</a>	<a href="#">5.3.</a>	Connection Handling . . . . .
<a href="#">17</a>	<a href="#">6.</a>	Vendor . . . . .
<a href="#">17</a>	<a href="#">6.1.</a>	Order Information . . . . .
<a href="#">17</a>	<a href="#">6.2.</a>	Ownership Validation . . . . .
<a href="#">17</a>	<a href="#">7.</a>	Configlet . . . . .
<a href="#">17</a>	<a href="#">7.1.</a>	Overview . . . . .
<a href="#">17</a>	<a href="#">7.2.</a>	Data Model . . . . .
<a href="#">18</a>	<a href="#">7.3.</a>	Signature . . . . .
<a href="#">19</a>	<a href="#">7.4.</a>	Encryption (optional) . . . . .
<a href="#">19</a>	<a href="#">7.5.</a>	YANG Module . . . . .
<a href="#">19</a>	<a href="#">8.</a>	Security Considerations . . . . .
<a href="#">21</a>	<a href="#">8.1.</a>	Immutable storage for trust anchors . . . . .
<a href="#">22</a>	<a href="#">8.2.</a>	Substitutions . . . . .
<a href="#">22</a>	<a href="#">8.3.</a>	Confidentiality . . . . .
<a href="#">22</a>	<a href="#">8.4.</a>	Entropy loss over time . . . . .
<a href="#">23</a>	<a href="#">8.5.</a>	Serial Numbers . . . . .
<a href="#">23</a>	<a href="#">9.</a>	IANA Considerations . . . . .
<a href="#">23</a>	<a href="#">9.1.</a>	ZeroTouch Information DHCP Option . . . . .
<a href="#">23</a>		

<a href="#">9.2.</a>	Media Types for Images and Configurations . . . . .	23
<a href="#">10.</a>	Acknowledgements . . . . .	23
<a href="#">11.</a>	References . . . . .	24
<a href="#">11.1.</a>	Normative References . . . . .	24
<a href="#">11.2.</a>	Informative References . . . . .	24
<a href="#">Appendix A.</a>	Examples . . . . .	24
<a href="#">A.1.</a>	Signed Configlet . . . . .	25
<a href="#">A.2.</a>	Signed Encrypted Configlet . . . . .	28
<a href="#">Appendix B.</a>	Change Log . . . . .	28
<a href="#">B.1.</a>	ID to 00 . . . . .	28
<a href="#">B.2.</a>	00 to 01 . . . . .	28

## [1.](#) Introduction

### [1.1.](#) Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

### [1.2.](#) Objectives

A fundamental business requirement is to reduce operational costs where possible. Deploying new IP-based devices is many times one of the largest costs in running a network, as sending trained specialists to each site to do an installation is both cost prohibitive and does not scale.

Both networking vendors and standard bodies have tried to address this issue, with varying levels of success. For instance, the Broadband Forum TR-069 specification [[TR069](#)] relies solely on DHCP for NMS discovery, but this can only work in environments where the DHCP server is locally administered, which is not the case when the device is connected to an ISP's network. In another example, some network vendors have enabled their devices to load an initial configuration from removable storage media (e.g., a USB flash drive), but not all devices have such ports.

The solution presented herein, ZeroTouch, enables a device to securely obtain an initial configuration from the network without

any

operator intervention. The discovered configuration initiates the device to "call home" using either the SSH or TLS, as described in [[NETCONF-REVERSE-SSH](#)] and [[RFC5539bis](#)] respectively.

### **1.3. Use Cases**

- o Connecting to a remotely administered network

This use-case involves scenarios, such as a remote branch office or convenience store, whereby the device connects to an ISP's network. In this case, the device receives only generic networking settings (address, netmask, gateway, DNS servers, etc.) provided by the ISP, with no site-specific customizations, such that the device has no recourse but to reach out to the presumably insecure network for its initial configuration.

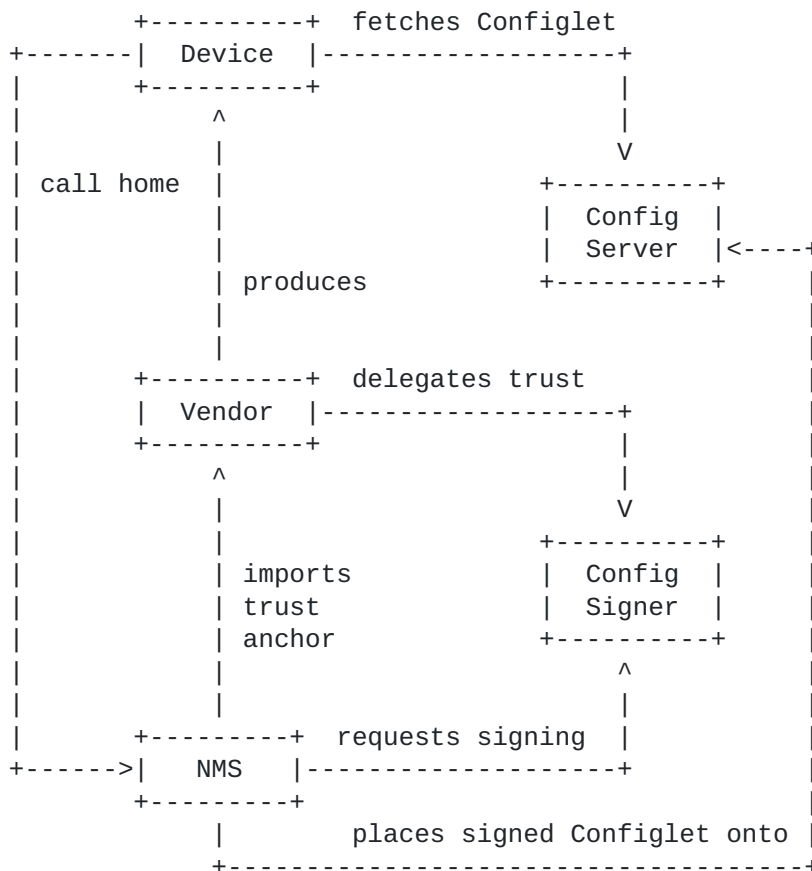
- o Connecting to a locally administered network

This use-case covers all other scenarios and differs only in that the device may additionally receive some site-specific information to guide its call home process, which could then direct it to a local server for its initial configuration. If no site-specific information is provided, or the device is unable to use the information provided, it can then reach out to network just as it would for a remotely administered network.





**1.4. Actors and Roles**



Though not represented as a box in the diagram, the Configlet is also a first-class object in the solution.

o Configlet

A Configlet is an XML document that, when loaded onto a device, configures the device to initiate a call home connection to a deployment specific NMS, as well as set a local administrator account for the NMS to log into. The Configlet is signed and optionally encrypted. More information about Configlets is in [Section 7](#).

o Configuration Server

A Configuration Server hosts configurations to be downloaded over a network. Configuration Servers can be deployed either on the locally administered network or on some external network



(e.g., the Internet). Configuration Servers are known to devices in the form of a URI, which can be either preconfigured or dynamically discovered. More information about Configuration Servers is in [Section 2](#).

- o Configuration Signer

A Configuration Signer is an entity that the device's vendor has delegated the signing function to. A Configuration Signer only needs to ensure that the requestor is the rightful owner of the device to which a configuration is destined. A Configuration Signer may be site-specific or an external entity. More information about Configuration Signers is in [Section 3](#).

- o Device

The device is the networking entity that initiates ZeroTouch, whenever booting with its factory default settings. The device is preconfigured with a secure device identity, for Configuration Servers URIs, and certificates for Configuration Signers and Configuration Servers it trusts by default. A device may dynamically discover additional URIs and certificates from a locally-administered network. More information about Devices is in [Section 4](#).

- o Network Management System

The NMS is the deployment-specific system that devices initiate their call home connections to. The NMS must be configured with vendor-specific trust anchors and unique device identifiers. The administrators of the NMS system interact with Configuration Signer and Configuration Server systems to stage the the device configurations. More information about Network Management Systems is in [Section 5](#).

- o Vendor

Vendors manufacture the devices with secure device identities and preconfigured Configuration URIs, and Configuration Signer certificates. Vendors are the de facto Configuration Signer for the devices it manufactures, but may delegate that role to external Configuration Signers. More information about Vendors is in [Section 6](#).



## **2. Configuration Server**

A Configuration Server is the entity hosting configurations that can be downloaded over a network. This section describes the service interface a Configuration Server must implement as well as what's needed for transport security.

### **2.1. Service Interface**

Configuration Servers are known to devices in the form of a URI. Configuration Servers MUST support the URI schemes "https" and "http". Other URI schemes are not supported.

When accessing a Configuration Server, the device appends its unique device identifier (UID) to the URI. The unique identifies MUST be the same as the identifier stored within the device's IDevID certificate.

For instance, if the URI were:

```
http://example.com/zerotouch/devices/  
https://example.com/zerotouch?id=
```

then the device would try to access:

```
http://example.com/zerotouch/devices/<uid>  
https://example.com/zerotouch?id=<uid>
```

When accessing the Configuration Server, the HTTP Accept-Type MUST be set to either "application/zerotouch-config" or "application/zerotouch-bootimage". Please see [Section 9.2](#). A wildcard Accept-Type (e.g., \*/\*) SHOULD default to "application/zerotouch-config".

### **2.2. Interactive Interface**

The Configuration server SHOULD to provide some user-facing interface to enable to the end-user to provide a Configlet and, optionally, an bootimage file. How the Configlet and bootimage file are provided to the Configuration Server is outside the scope of this document.



### **2.3. Transport Security**

As described in [Section 3](#), configurations MUST be signed and MAY be encrypted. As such, transport-level security is not needed to assure

authenticity or confidentiality of the configuration itself. However, transport-level security enables devices to authenticate the

Configuration Server and also extends confidentiality to the application-level protocol. Therefore, it is RECOMMENDED for Configuration Servers to support transport-level encryption.

If a Configuration Server uses X.509-based encryption, then its X.509

certificate MUST have a chain of trust to a trust anchor known to devices (see [Section 4.2](#). More specifically, the Configuration Server MUST possess all the intermediate certificates leading to the trust anchor.

When a Configuration Server negotiates encryption with the device, it

provides the chain of certificates, from its own to, but not including, the trust anchor. Including the trust anchor's certificate is unnecessary since the device MUST be pre-provisioned with it. Devices need the chain of certificates to be passed so they

can validate the server using only a list of Configuration Server trust anchors.

### **2.4. Expiration Policy**

An expiration policy is needed to limit how long a Configuration Server needs to retain a configuration and, in turn, how many configurations it might need to retain at a given time.

It is expected that Configuration Servers will enable retention information to be given at the same time as when the configuration is

provided to it. Options should be temporal in nature, not based on access counts, so as to thwart a DoS attack whereby the configuration

is accessed by an entity other than the device. Configuration Servers SHOULD put a limit on the maximum amount of time it will hold

onto a configuration before purging it, even if the configuration had never been accessed.

### **2.5. Troubleshooting and Auditing**

In order to facilitate troubleshooting and auditing, the Configuration Server SHOULD record into a log a record of the

various

Configlet download requests. This draft does not define what information should be kept or for how long.

Watson, et al.  
8]

Expires January 02, 2015

[Page



### **3. Configuration Signer**

#### **3.1. Overview**

A Configuration Signer MUST be able to sign configurations. This function requires the Configuration Signer be able to authenticate that the requestor is the true owner of the device, as identified within the contents of the configuration being signed.

The user interface a Configuration Signer provides to perform its role is outside the scope of this document. However, in order to meet operational expectations, the time it takes to respond to a request should be as expeditious as possible.

A Configuration Signer does not need to retain a configuration after signing it. The Configuration Signer SHOULD retain an audit log for indemnification purposes.

#### **3.2. Signing Configurations**

A Configlet Signer MUST have an X.509 certificate with Key Usage capable of signing data (digitalSignature) and be signed by a certificate authority having a chain of trust leading to a trust anchor known to the devices loading its Configlets. The Configlet Signer MUST possess all intermediate certificates leading to its trust anchor.

When a Configlet Signer signs a Configlet, it attaches both the signature and the chain of X.509 certificates, including its own, but not necessarily including the trust anchor's certificate. This chain of certificates is needed so a device can validate a Configlet using only the Configlet Signer trust anchors known to it.

#### **3.3. Optional Encryption**

A Configuration Signer MAY optionally encrypt configurations prior to signing them. This function requires the Configuration Signer know the device's unique public key, as encoded within its secure device identity certificate.

#### **3.4. Delegation of Trust**

A device's vendor is the root of trust for all of its devices. That is, the vendor's devices implicitly trust the vendor for such things as software images, subscription updates, and licenses. As such, the vendor is the ultimate Configuration Signer for its devices.



However, both vendors and its customers may prefer a this role be performed by another entity. For instance, a vendor may not want this role due to it being outside its primary business function, and customers may not want the vendor to have this role for privacy reasons.

It is therefore provided that a vendor MAY delegate the Configuration

Signer role to other entities. Using X.509 certificates, the Vendor need only sign the delegate's certificate signing request (CSR), providing back to the delegate a signed X.509 certificate authenticating its ability to perform the signing function.

In order enable a delegate to fulfill its operational role, as described in [Section 3.1](#), the vendor MUST provide a mechanism that can be used to authenticate if a given requestor is the true owner of

a specific device. Additionally, to support Configuration Signers that want to encrypt configurations, the vendor MUST also provide a means for the Configuration Signer to know the public key for a given

device. How the vendor provides this information to Configuration Signers is outside the scope of this document.

### **[3.5. Delegation to a Specific Customer](#)**

The general expectation is that the Configuration Signer is an impartial 3rd-party. However, certain deployments may want to be able to perform the function for themselves. Yet without constraints, that deployment could sign configurations for devices that do not belong to it.

Resolving this concern is possible when 1) the deployment specific Configuration Signer's certificate is annotated with a customer identifier and 2) the devices sold to that customer have that same identifier encoded into their secure device identifier.

This entails the vendor augmenting its manufacturing process for these special devices, which would likely be sold directly to the customer, as opposed to through a sales channel. This takes extraordinary effort and likely only implemented for the most important customers, if at all.

## **[4. Device](#)**

### **[4.1. Overview](#)**

While the wholistic solution, ZeroTouch, involves a number of entities, a device being powered-on is the essential event that sets things in motion.







Devices supporting ZeroTouch MUST be pre-provisioned with one or more

URIs for Internet-based Configuration Servers. These URIs SHOULD be partitioned into one set that contains secure schemes (e.g.

https://)

and another set that contains insecure schemes (e.g., http://). The reason for partitioning the URIs is so all the secure schemes can be attempted before any of the insecure schemes (see [Section 4.3](#)).

When

using a secure scheme, the Configuration Server MUST be authenticated

using a trust anchor the device possesses. As each Configuration Server may use a different trust anchor, this generalizes to a list of Configuration Server trust anchor certificates.

In order to verify the signature on retrieved configurations, devices

supporting ZeroTouch MUST also possess the trust anchor for the Configuration Signer that signed the configuration. Generally, only the manufacturer's trust anchor is needed, as it can then delegate trust for 3rd-party Configuration Signers (see [Section 3.4](#)).

However, for various reasons, there may be a need for more than one root anchor and therefore this generalizes to a list of

Configuration

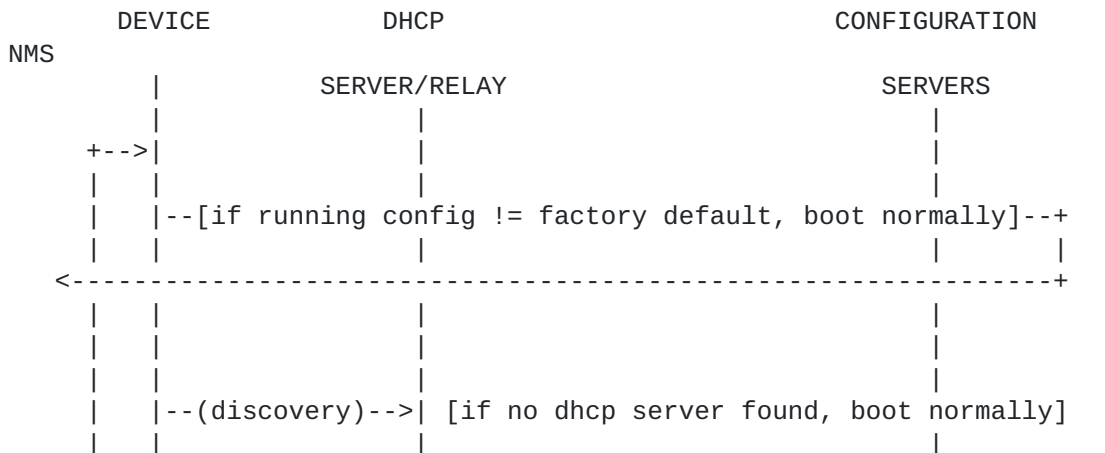
Signer trust anchor certificates.

Devices SHOULD ensure that the certificates for its trust anchors are

protected from external modification. It is for this reason that the

diagram shows the Configuration Signer and Configuration Server certificates in immutable storage. Similarly, per [\[Std-802.1AR-2009\]](#), the IDevID private key shall be stored confidentially and not available outside the DevID module, hence the diagram shows it is held within secure storage.

### 4.3. Boot Sequence



```
| | +---(offer)---| | | | |
| | | | | | |
| | +--[add any listed config servers to built-in list]--+ | |
| | | | | | |
| |<-----+-----+ | |
```







If a DHCP server provides both the ZeroTouch Information and the vendor specific information DHCP options, then the ZeroTouch Information option MUST be processed first. After exhausting all ZeroTouch options without being able to call home, a device MAY then process the information provided by the legacy DHCP options.

The ZeroTouch Information option [Section 9.1](#) provides a set of Configuration Server URIs. If returned by the DHCP server, the device MUST append each URI to the end of one of its two sets of Configuration Server URIs, depending on if the URI's scheme is secure or not. URIs added this way MUST remain distinguishable from those URIs the device was shipped with, for reasons discussed next.

The device then iterates over its two sets of Configuration Server URIs. The device MUST first try all the URIs from the set having secure schemes before trying any of the URIs from the set having insecure schemes. For each URI, until a match is found and successfully loaded, the device attempts to initialize itself from the URI. If the URI uses a secure scheme (e.g., https), the device MUST validate the Configuration Server's certificate using one of its Configuration Server trust anchors. If the device is unable to verify the server's certificate, the device MUST skip that URI. If the device reaches the end of all its URIs without finding a usable match, it SHOULD continue its normal boot sequence using its factory default configuration.

When the device is accessing a Configuration Server URI that it was shipped with (i.e. not discovered while initializing its networking), it MUST do so by appending its GUID to the URI string and using the Accept-Type "application/zerotouch-config", as described in [Section 2](#). For URIs discovered via the ZeroTouch Information option, the device MAY also try the raw URI after trying the permutation using its GUID.

If the Configuration Server returns a configuration, the device MUST first verify it before use. Configuration verification entails both verifying the configuration's signature using the device's list of Configuration Signer trust anchors, and also verifying that the unique identifier within the Configlet matches the device's unique identifier.

Once the configuration is authenticated, the device MUST compare its software image version with the expected version specified within the configuration. If there is a mismatch, the device MUST download the correct image version from the Configuration Server, by appending its GUID to the Configuration Server's URI string and using the Accept-

Type "application/zerotouch-bootimage", as described in [Section 2](#).  
For URIs discovered via the ZeroTouch Information option, the device

MAY also try both the raw URI after trying the permutation using its GUID. Once the image has been downloaded, the device MUST install it and reboot, still with the factory default settings configured, so that ZeroTouch restarts when the device comes back up.

If the device is running the correct software image version, it merges the Configlet's contents into its running configuration. This step effectively modifies the device so that it is no longer having its factory default setting. However, since the Configlet configured the device to "call home," upon entering its normal operating mode, the device immediately begins trying to establish a call home connection, as specified by the Configlet.

If configured to establish a SSH connection, the the device MUST use its IDevID and associated intermediate X.509 certificates as its host key per [RFC 6187](#) [[RFC6187](#)]. If configured to establish a TLS connection, the device MUST use its IDevID and associated intermediate X.509 certificates as its server-side certificate for the TLS connection.

In order to facilitate troubleshooting, the device SHOULD record into a log information relating to its stepping through the ZeroTouch sequence of steps. This draft does not define any specific log messages, for instance, for Syslog or SNMP.

## **5. Network Management System (NMS)**

### **5.1. Overview**

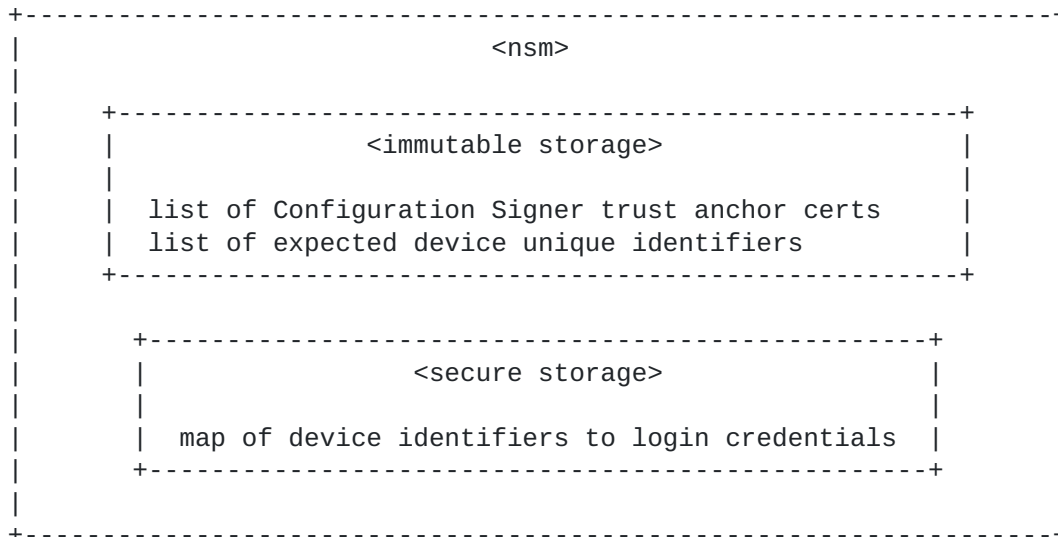
The NMS is the ultimate destination of ZeroTouch for a device. It is the NMS's network address configured in the Configlet. The device will initiate a call home connection to the NMS, using either a SSH or TLS, as configured by the Configlet loaded.

Watsen, et al.  
15]

Expires January 02, 2015

[Page

## 5.2. Precondition



In order to authenticate the device, the NMS MUST possess the X.509 certificate for the trust anchor leading to the device's entity certificate. The NMS uses this certificate to validate the server-certificate the device presents during SSH or TLS transport negotiation. Because an NMS may interoperate with multiple vendors, and a vendor may have more than one trust anchor for signing its devices IDevID certificates, this generalizes into the NMS needing a list of trust anchor certificates. This certificates SHOULD be stored in a way that prevents tampering, which is why they are shown in immutable storage in the diagram.

In order for the NMS to validate that the specific device connecting to it is expected, the MUST have a list of unique device identifiers that it can use to validate the device's IDevID certificate with. The list SHOULD be protected from external modification, which is why it is shown in immutable storage in the diagram. In order for the NMS to know the unique identifiers, device manufacturers will need to provide a mechanism to convey this information to its customers. This draft not specify a format for this information exchange.

In addition to authenticating the device, the NMS must also authenticate itself to the device. How this is done is deployment specific, but generalizes to the NMS needing to have login credentials for each device. These credentials will entail knowing a secret (e.g., password, private key). For this reason the diagram shows the NMS storing a map of device credentials in secure storage.





### **5.3. Connection Handling**

When receiving a NETCONF call home connection from a device, the NSM completes the connection as specified in the SSH [[NETCONF-REVERSE-SSH](#)] and TLS [[RFC5539bis](#)] drafts.

## **6. Vendor**

### **6.1. Order Information**

In order for a Vendor's customers to preconfigure their NMSs with what devices are expected, as well as to know how to set the "unique-identifier" field within a Configlet when requesting a signing, Vendors need to provide a mechanism for customers to obtain the unique identifier value for the devices they have ordered. For instance, customers could receive emails containing shipping information for their devices.

Additionally, to facilitate workflows where the devices are initially received by a customer-specific warehouse, or moved after having been unboxed, it is ideal for the unique identifier to be easily tracked through labels affixed to the device as well as the box it is packaged in. A device's serial number is commonly treated this way and would be suitable for this purpose, so long as it is directly related to its IDevID identity.

### **6.2. Ownership Validation**

In order for Configuration Signers to validate that a requestor is the true owner of a device (i.e. its IDevID identity), Vendors need to provide a mechanism enabling a near real-time lookup. The interface used to implement this lookup is outside the scope of this document.

## **7. Configlet**

### **7.1. Overview**

A Configlet is an XML file, containing specific YANG-defined configuration, that has been signed by a trusted signer known to the device (e.g., the device's manufacturer).



The Configlet data-model, defined by the YANG module in this document (see [Section 7](#)), is just enough to configure a local user account and either reverse-SSH or reverse-TLS. More specifically, this data-model is a subset of what's defined in `ietf-system` and `ietf-netconf-server` YANG models. This focused data-model is consistent with the common use-case of having the NMS push a full configuration to a device when it calls home.

The signature on the Configlet is enveloped, meaning that the signature is contained inside the XML file itself. The signature block also contains the X.509 certificate of the Configlet Signer and its chain of trust.

Once a device authenticates the signature on a Configlet and matches the unique identifier (e.g., serial number) within the Configlet, it merges the configuration contained in the Configlet into its running datastore.

## [7.2.](#) Data Model

```
module: ietf-netconf-zerotouch
  +-rw configlet
    +-rw target-requirements
      | +-rw unique-identifier    string
      | +-rw software-version    string
    +-rw configuration
```

The Configlet's data model is no more than a wrapper around a header (i.e. `<target-requirements>`) and a payload (i.e. `<configuration>`).

The `<target-requirements>` element contains information that MUST be validated by the device prior to processing the `<configuration>` element. Specifically, it contains:

- o unique-identifier

- The unique-identifier field is used to ensure that the Configlet is loaded onto the targeted device and no other. This field is also used by the Configuration Signer, when ensuring the requestor is the true owner of the device. The value MUST be the same as the 'subject' field in the device's DevID credential, as specified by [section 7.2.8](#) in IEEE Std 802.1AR-2009.

- o software-version

- The software-version field is used to ensure the device is running the right software version prior to loading the



configuration (e.g., 14.1R2.5). If the device finds that it is not running the correct version of software, it can pull the correct version from the Configuration Server.

The <configuration> element contains the configuration that is to be committed to the device's running datastore. This element uses the "anyxml" type, enabling it to contain either vendor-specific or standards-based data models. When using standard models, in order to complete a call home connection, only the following is needed:

- o The "authentication" subtree from "ietf-system", defined in [draft-ietf-netmod-system](#).
- o If TLS is supported, everything from "ietf-system-tls-auth", defined in [draft-ietf-netconf-server-model](#).
- o The "call-home" subtree from "ietf-netconf-server", defined in [draft-ietf-netconf-server-model](#).

### **7.3. Signature**

All Configlets MUST be signed by a Configuration Signer in order to be authentic. Devices MUST reject any Configlet that is either unsigned or having an invalid signature. Configlets are signed using

the W3C standard "XML Signature Syntax and Processing" [[XMLSIG](#)].

The

entire contents of the Configlet MUST be signed. The signature block

must also include the Configlet Signer's certificate and any intermediate certificates leading to a Configlet Signer trust anchor.

A signed Configlet example is in section [Appendix A.1](#).

### **7.4. Encryption (optional)**

Configlets MAY optionally be encrypted prior to being signed. Encrypting the Configlet provides confidentiality for the Configlet's

contents without relying on transport-level security. Configlets are

encrypted using the W3C standard "XML Encryption Syntax and Processing" [[XMLENC](#)]. The entire contents of the Configlet MUST be encrypted. An encrypted Configlet example is in section [Appendix A.2](#).

### **7.5. YANG Module**

Following is the YANG module for the Configlet:

```
module ietf-netconf-zerotouch {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-zerotouch";  
    prefix "zerotouch";
```

Watsen, et al.  
19]

Expires January 02, 2015

[Page

organization

"IETF NETCONF (Network Configuration) Working Group";

contact

"WG Web: <<http://tools.ietf.org/wg/netconf/>>

WG List: <<mailto:netconf@ietf.org>>

WG Chair: Mehmet Ersue  
<<mailto:mehmet.ersue@nsn.com>>

WG Chair: Bert Wijnen  
<<mailto:bertietf@bwijnen.net>>

Editor: Kent Watsen  
<<mailto:kwatsen@juniper.net>>";

description

"This module contains a collection of YANG definitions for configuring NETCONF zerotouch.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

// RFC Ed.: replace XXXX with actual RFC number and

// remove this note

// RFC Ed.: please update the date to the date of publication

```
revision "2014-07-01" {
  description
    "Initial version";
  reference
    "RFC XXXX: A YANG Data Model for NETCONF ZeroTouch Configlet";
}
```

```
container configlet {
```

```
  description
```

```
    "Top-level container for ZeroTouch configuration objects.";
```





```
    container target-requirements {
      description
        "Specifies requirements for device this is loaded onto";
      leaf unique-identifier {
        type string;
        mandatory true;
        description
          "The device MUST have this unique identifier. The value
          MUST be the same as the 'subject' field in the device's
          DevID credential, as specified by section 7.2.8 in
          IEEE Std 802.1AR-2009.";
      }
      leaf software-version {
        type string;
        mandatory true;
        description
          "The device MUST must be running this version of software.
          The value for this field is device-specific, but it MUST
          be an exact match (e.g., 14.1R2.5)";
      }
    }
  }
  anyxml configuration {
    mandatory true;
    description
      "The configuration to be committed to the device's running
      datastore. The configuration MUST be valid for the
      target
      device. Device's supporting ZeroTouch SHOULD at least
      support both the following standard data-models:

          ietf-system           // the authentication container
          ietf-system-tls-auth // everything, if TLS supported
          ietf-netconf-server  // the call-home container

      These three data models contain everything needed to
      support NETCONF call home using either SSH or TLS.";
  }
}
```

## 8. Security Considerations



### **8.1. Immutable storage for trust anchors**

Devices SHOULD ensure that all its trust anchor certificates, including those for the Configuration Signer and Configuration Server, are protected from external modification. It is for this reason that the diagram in [Section 4.2](#) shows them in immutable storage.

However, it may be necessary to update these certificates over time (e.g., the vendor wants to delegate trust to a new CA). It is therefore expected that devices MAY update these trust anchors when needed through a verifiable process, such as a software upgrade using signed software images.

### **8.2. Substitutions**

It is generally not possible to substitute a Configlet created for a different device, since devices assert that the Configlet contains their unique identifier (e.g., serial number).

However, it is possible to substitute a Configlet created for a device with a different Configlet created for the same device. Generally, unless imposed by the Configuration Signers, there is no limit to the number of Configlets that may be generated for a given device. This could be resolved, in part, by placing a timestamp into the Configlet and ensuring devices do not load Configlets older than some amount, but this requires the devices have an accurate clock when validating a Configlet and for Configuration Signers to not sign a Configlet when another Configlet is still active.

### **8.3. Confidentiality**

This draft allows devices to use insecure schemes when doing a Configuration Server lookup. This is deemed acceptable because the Configlet is tamper-proof, since it MUST be signed, only confidentiality is lost.

Confidentiality of a Configlet's contents is assured when either the Configlet is encrypted or when a secure scheme is used when accessing the Configuration Server.

Some confidentiality is lost when an insecure scheme is used to access a Configuration Server, as then the device's unique identifier is in the clear.



Given the fairly regular format for unique identifiers, it is possible that an adversary to guess unique identifiers and access a device's Configlet. Configlets that have been encrypted do not disclose any confidential information.

#### **8.4. Entropy loss over time**

[Section 7.2.7.2](#) of the IEEE Std 802.1AR-2009 standard says that IDevID certificate should never expire (i.e. having a notAfter 99991231235959Z). Given the long-lived nature of these certificates, it is paramount to use a strong key length (e.g., 512-bit ECC). Vendors SHOULD deploy Online Certificate State Protocol (OCSP) responders or CRL Distribution Points (CDP) to revoke certificates in case necessary.

#### **8.5. Serial Numbers**

This draft mentions using the device's serial number as its unique identifier in its IDevID certificate. This is because serial numbers are ubiquitous and prominently contained in invoices and on labels affixed to devices and their packaging. That said, serial numbers many times encode revealing information, such as the device's model number, manufacture date, and/or sequence number. Knowledge of this information may provide an adversary with details needed to launch an attack. To address this concern, the certificate could contain the hash of the serial number instead, which the NMS could also compute, but doing so is much less intuitive and raises questions if it is just security through obscurity.

### **9. IANA Considerations**

#### **9.1. ZeroTouch Information DHCP Option**

TBD, but it essentially returns a list of URIs.

#### **9.2. Media Types for Images and Configurations**

TBD, but in accordance with [RFC 6838](#), the draft registers: application/zerotouch-configlet and application/zerotouch-bootimage

### **10. Acknowledgements**

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): David Harrington, Dean Bogdanovic, Martin Bjorklund, Wes Hardaker, Russ Mundy, Reinaldo Penno, Randy Presuhn, Juergen Schoenwaelder.

Watsen, et al.  
23]

Expires January 02, 2015

[Page

Special thanks goes to Russ Mundy and Wes Hardaker for brainstorming the original I-D's solution during the IETF 87 meeting in Berlin.

## **11. References**

### **11.1. Normative References**

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels ", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols ", [RFC 3365](#), August 2002.
- [RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol ", [RFC 4252](#), January 2006.
- [RFC5539bis] Badra, M. and A. Luchuk, "Using the NETCONF Protocol over Transport Layer Security (TLS) ", [RFC 5539](#), March 2011.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication ", [RFC 6187](#), March 2011.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "NETCONF Configuration Protocol", [RFC 6241](#), June 2011.
- [NETCONF-REVERSE-SSH] Watsen, K., "NETCONF over SSH Call Home", April 2014.
- [Std-802.1AR-2009] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009.
- [XMLSIG] , "XML Signature Syntax and Processing", April 2013.
- [XMLENC] , "XML Encryption Syntax and Processing", April 2013.

### **11.2. Informative References**

- [TR069] The Broadband Forum, ., "TR-069 Amendment 3, CPE WAN Management Protocol ", November 2010.

## **Appendix A. Examples**





### [A.1.](#) Signed Configlet

This example illustrates a Configlet configuring both a local user account and call home using SSH. This Configlet includes both the Configuration Signer's certificate as well as an Intermediate certificate. Note that '\' characters have been added for formatting reasons.

```
<?xml version="1.0"?>
<configlet xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-zerotouch">

  <target-requirements>
    <unique-identifier>0123456789</unique-identifier>
    <software-version>14.1R3.5</software-version>
  </target-requirements>

  <configuration>

    <!-- from ietf-system.yang -->
    <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
      <authentication>
        <user>
          <name>admin</name>
          <ssh-key>
            <name>admin's rsa ssh host-key</name>
            <algorithm>ssh-rsa</algorithm>
            <key-
data>AAAAB3NzaC1yc2EAAAADAQABAAQDeJMV8zrtsi8CgEsRC
                jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/
02Mwj

E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC
                WAw1l0r9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/
fL3ZS5G76Fy6sA5
                vg7SLqQFPjXXft2CAhin8xwYRZy6r/
2N9PMJ2Dnepvq4H2DKqBIe340jWq
                EIuA7LvEJYq14unq4Iog+/
+CiumTkmQIWRgIoj4FCzYk09NvRE6f0SLLf6
                gakWVOZZgQ8929uWjCWlG1qn2mPibp2Go1</key-data>
            </ssh-key>
            <!--<password>$1$salt$hash</password>-->
          </user>
        </authentication>
      </system>

    <!-- from ietf-netconf-server.yang -->
    <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
server">
      <ssh>
        <call-home>
```

```
<applications>
  <application>
    <name>config-mgr</name>
    <description>
      This entry requests the device to periodically
```

Watsen, et al.  
25]

Expires January 02, 2015

[Page

```
        connect to the Configuration Manager application
    </description>
    <servers>
        <server>
            <address>config-mgr1.example.com</address>
        </server>
        <server>
            <address>config-mgr2.example.com</address>
        </server>
    </servers>
    <connection-type>
        <periodic>
            <timeout-mins>5</timeout-mins>
            <linger-secs>10</linger-secs>
        </periodic>
    </connection-type>
    <reconnect-strategy>
        <start-with>last-connected</start-with>
        <interval-secs>10</interval-secs>
        <count-max>3</count-max>
    </reconnect-strategy>
    <host-keys>
        <host-key>
            <name>ssh_host_key_cert</name>
        </host-key>
        <host-key>
            <name>ssh_host_key_cert2</name>
        </host-key>
    </host-keys>
    </application>
</applications>
</call-home>
</ssh>
</netconf-server>

</configuration>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference>
      <Transforms>
        <Transform
          Algorithm=\
            "http://www.w3.org/2000/09/xmldsig#enveloped-signature" /
        </Transforms>
```

>



```
<DigestMethod
  Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<DigestValue>2x1FdlVifb1snGBLJuEZYrLjSUQ=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>
HUx3S7TZxGJGUhazWGRSB9CBMZ0T+tTrB1f0nTcKi9wU4U0nSw5KMWDvOVwc6ldM
UIOJiUjigWhSkn+vVwSwz6qy7LTYIywNcxDyghMvmMXfoRXETpL+qCDxribMi4Vw
mVhEw1oe83kjt7W/0DJUE7FFKRuHPjy9EgxpQX/7WdKSK+4f2uYkSpq2UumW3DIU
LeK9vNRVQBbhmCF3zZWANmwKH5V4WeQimwWE497AeSYwgSImSetADI0NvvXfBZjx
JqzFEaYLNz8IB0ZVY+w14s1RZbn7YmxhN1R3q52wvHjR2SylR/Z5BpIhYoDeKoD
HMQMf3HZL06Hm5S8r8rgGg==</SignatureValue>
<KeyInfo>
  <X509Data>
    <X509Certificate>
MIIFKjCCBBKgAwIBAgIBAJANBgkqhkiG9w0BAQsFADAwMRMwEQYDVQQKFApUUE1f
VmVuZG9yMRkwFwYDVQQDFBFBKdW5pcGVyX1hYWfYX0NBMB4XDTEzMTA0MjE1MjE1
MFoXDTE0MTA0MjE1MjE1MFowKzETMBEGA1UEChQKVFBjZjE1ZjE1ZjE1ZjE1ZjE1
AxQly2hpcF8wMDAwMDEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDf
4hyWqFsf801sZYJQBj0PB4cHmlnPN0s9pv3QCCB1Pz1YhfcDOygVmghzZjPY+t7q
ZTjPs/E8n5X4dd0DKR80uc4Mwmc40Pz2HAW6GQ2mo+eUYzXUqQFbi3EkqrzddZk
gRi6vuadMkAcJH8ugYR+cbw/LlpXhIy2A5fU4JP7Y911wABTbK8eGhF9cvGxBYR
+KqZJycoV6aaIvD/0N01CNSaGeAJXXxXWoRF5E6HVKsolTHPPdi+40BmYrCuuWy6
1ybCIP5uZZ70za4j0n/fPb6SEqEa0I1zUEWlFQMZYsBC1NY5TzWHNgQ5dPJ02qgx
PONwnLIsx46D1AzlpFpXAgMBAAGjggJSMIICTjAMBGNVHRMBAf8EAJAAMIGTBgNV
HSABAF8EgYgwgYUwgYIGC2CGSAGG+EUBBY8BMHMwOQYIKwYBBQUHAgEWLWh0dHA6
Ly93d3cudmVyaXNpZ24uY29tL3JlcG9zaXRvcnkvaW5kZXguaHRtbDA2BggrBgEF
BQcCAjAqGihUQ1BBIFRydXN0ZWQUGXhdGZvcmt0gTW9kdWx1IEVuzG9yc2VtZW50
MIHXBGNVHSMEGc8wgcyAFCHd7bYICEQX3QXR30ixhpg7bjmoYgwpIGtMIGqMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcn5pYTESMBAGA1UEBxMjU3Vubn12
YWx1MRkwFwYDVQQKFApUUE1fVmVuZG9yMRkwFwYDVQQDFBFBKdW5pcGVyX1hYWfYX0NB
Y2F0ZV9Jc3N1YW5jZTEZMbcGA1UEAxQKVFBjZjE1ZjE1ZjE1ZjE1ZjE1ZjE1ZjE1
SIB3DQEJARY0Y2FAanVuaXB1ci5jb22CAQEWcQYDVR0fBGowaDBmoC6gLIYqaHR0
cDovL2Nybc5qdW5pcGVyLm5ldD9jYT1KdW5pcGVyX1hYWfYX0NB0jSkMjAwMRMw
EQYDVQQKFApUUE1fVmVuZG9yMRkwFwYDVQQDFBFBKdW5pcGVyX1hYWfYX0NBMFsG
A1UdEQEB/wRRME+kTTBLMQswCQYDVQQGEwJVUzEYMBYGA1UEChMPTXkgT3JnYW5p
emF0aw9uMRAwDgYDVQQLEwdNeSBVbm10MRAwDgYDVQQDEwdNeSB0YW11MA0GCSqG
SIB3DQEBcWUAA4IBAQCsvFA9008E4p/8ohBYQRezVaWidTHCTM1sdAoe1jlrFX
xqwcQEGVT3BpzWn8w2r+iK0KLQkw64os0KKL0RIIjMcmJ2RukqH/R0M8Air4+Im
iWI3xv+HzVRsJIrcRT2tzxbchU/i/LQiwhteUEZ9sZbHKyLQe9x9HgByM05if0Gh
z2dcb7AWNlo7nJtRBmx0v9iim2kktqGMuXgBzlnMMabqHMb4L+vjww2Wn5nNYbr/
oXq4fa01MGQyvrPAE0wL3ZxcaqKHvTn9coBLhpP3nQIEV+V+PngQjtBmwdkjiJ5
feDp86jGN6348H+z9CzXUSby0n6utIxN0SvVESxx</X509Certificate>
    <X509Certificate>
MIIEExtCCA62gAwIBAgIBATANBgkqhkiG9w0BAQsFADCBqjELMAkGA1UEBhMCMVVMx
EzARBGNVBAgTCKNhbg1mb3JuaWExEjAQBGNVBAcTCVN1bm55dmFsZTEZMbcGA1UE
ChQQSnVuaXB1cl90ZXR3b3JrczEdMBSGA1UECjQUQ2VydGlmawNhdGVfSxNzdWfu
Y2UxGTAXBGNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWdMnh
```









Major structural update; the essence is the same. Most every section was rewritten to some degree.

Added a Use Cases section

Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram

Removed support for physical presence or any ability for Configlets to not be signed.

Defined the ZeroTouch Information DHCP option

Added an ability for devices to also download images from Configuration Servers

Added an ability for Configlets to be encrypted

Now Configuration Servers only have to support HTTP/S - no other schemes possible

#### Authors' Addresses

Kent Watsen  
Juniper Networks

E-Mail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

Stephen Hanna  
Juniper Networks

E-Mail: [shanna@juniper.net](mailto:shanna@juniper.net)

Joe Marcus Clarke  
Cisco Systems

E-Mail: [jclarke@cisco.com](mailto:jclarke@cisco.com)

Mikael Abrahamsson  
T-Systems

E-Mail: [mikael.abrahamsson@t-systems.se](mailto:mikael.abrahamsson@t-systems.se)

